Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

SENDAS: Scalable ENrichment for mobility DAta Sets

Henrique Santana
Universidade Federal de Viçosa
Germano Santos
Universidade Federal de Viçosa
Fabrício Silva
fabricio.asilva@ufv.br
Universidade Federal de Viçosa
Thais Silva
Universidade Federal de Viçosa

Research Article

Keywords:

Posted Date: May 11th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-9371914/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

SENDAS: Scalable ENrichment for mobility DAta Sets

Henrique Santana[1], Germano Santos[1], Fabrício Silva[1*], Thais Silva[1]

[1]Laboratório de Inteligência em Sistemas Pervasivos e Distribuídos (NESPeD-Lab), UFV, Rodovia LMG, Florestal, 100190, MG, Brazil.

*Corresponding author(s). E-mail(s): fabricio.asilva@ufv.br;
Contributing authors: henrique.s.santana@ufv.br;
germano.santos@ufv.br; thais.braga@ufv.br;

Abstract

Recent technological advancement and popularization have grown the availability of geo-referenced mobility data, collected from different sources. Extracting knowledge from such large amounts of data is an active research and industrial issue, with a diverse range of applications. However, existing solutions currently lack scalability. Moreover, some important features are not available in such solutions. To cover this gap, we introduce SENDAS – Scalable ENrichment for mobility DAta Sets – a new framework enabling parallel and distributed execution for mobility analysis techniques. We present definitions extending and improving the efficiency of state-of-the-art methods to extract stay points, areas of interest, semantic motifs, origin-destination flows, among other metrics. The performance results reveal that it was possible to speed up the processing time up to 8.6 times when compared to existing solutions in large-scale datasets. In addition, we present a case study of SENDAS by comparing the mobility patterns of millions of users during the COVID-19 pandemic period.

1 Introduction

Human mobility is a broad research field, comprehending multiple disciplines and being of interest to diverse applications, like urban planning [5], population migration analysis [37], environmental impact studies [15], recommendation systems [35], among

others [25, 43]. Besides that, historical mobility data has been used by researchers and industry to understand and characterize human mobility [3, 40, 46].

While in the past, detailed mobility data – also known as trajectory data – were scarce, with recent years of technological advancement and popularization for various purposes, its availability has been growing in an unprecedented scale and granularity [6, 10, 19, 44, 51]. The use of Call Detail Records (CDRs), GPS data, and social network geotagged posts are examples of data collection means that contributed to this change [1]. Thus, the problem now is the opposite, given that a large amount of such data poses challenges due to its big volume and heterogeneity.

Several solutions already exist to extract knowledge from mobility data, usually as Python or R libraries, such as *Moving-Pandas* [12], *Scikit-Mobility* [30], Trackintel [27], among others. However, they lack mainly in two aspects. First, some important features are not available, such as mobility motif extraction and semantic annotation, and multi-shape area of interest extraction. Secondly, and most importantly, they lack scalability.

In this work, we propose SENDAS[1] – Scalable ENrichment for mobility DAta Sets –, a framework that closes the aforementioned gaps and also provides implementations of state-of-the-art semantic enrichment algorithms and techniques for mobility analysis. SENDAS is built on top of Apache Spark [49] and uses Apache Sedona, formerly known as GeoSpark [47], to aid with geospatial queries.

The main contributions of this work can be summarized as:

- We derive definitions of trajectory and spatial representations that cover many requirements in terms of points and tiles, as well as different aspects of trajectories (Section 3.1);
- We improve a stay point inference method to handle large-scale datasets (Section 3.2.1);
- We propose a multi-shaped Area of Interest (AOI) inference and semantic annotation for large-scale datasets (Section 3.2.2);
- We propose a novel and efficient method to extract and annotate mobility motifs for large-scale datasets (Sections 3.2.3 and 3.2.4);
- We implement a method for origin-destination flow extraction for large-scale datasets (Section 3.2.5);
- We make SENDAS available to the community as an open-source library.

To demonstrate the benefits of SENDAS, we have conducted experiments under over 24 million mobility records. The performance results reveal that it was possible to speed up the processing time up to 8.6 times when compared to the existing solutions. Thus, we demonstrate that SENDAS can handle large amounts of mobility data. Moreover, to demonstrate the practical usage of SENDAS, we applied it in a case study to extract mobility characteristics from a real dataset with over 95 million records to compare how mobility patterns change semantically from 2021 to 2022 because of the COVID-19 pandemic.

The remainder of the text is organized as follows: Section 2 discusses related work found in the literature. The details and project design of SENDAS are described

[1] Open-source repository is available in http://github.com/NESPEDUFV/sendas

in Section 3. Section 4 shows an experimental and comparative evaluation of our proposed library. The practical case study is presented in Section 5. Finally, concluding remarks are given in Section 6, along with future directions and challenges.

2 Related Work

SENDAS is a framework built to extract mobility patterns from large-scale data. In addition to its big data characteristic, SENDAS also includes novel methods for stay point detection, extracting multi-shape areas of interest (AOI), motifs extraction, and semantic annotation. In this section, we present existing frameworks and libraries on the mobility subject and discuss how they differ from SENDAS. The related works specific to each mobility mining method can be found in the corresponding parts of Section 3.

Due to the lack of existing tools to handle large-scale mobility data, researchers need to develop their own methods, designing their packages to handle their particular problem [18, 33]. The many options of spatial solutions increase the difficulty of choosing the best library that fits the project requirements [13]. Moreover, the proprietary solutions are not available to the research community.

Several mobility data analysis and processing tools were developed to achieve specific goals for different purposes in mobility research. Hence, most libraries and frameworks handle small or medium-sized datasets, becoming an issue when dealing with big mobility data as presented in Table 1. The two existing solutions that handle big data are TrajMesa [24] and SPARTAN [34]. TrajMesa is a framework built on top of GeoMesa, a NoSQL distributed trajectory data system, to design a new efficient storage schema focused on reducing time spent on temporal, spatial range, similarity, and KNN queries. However, it offers only the stay-point inference method. On the other hand, SPARTAN is a framework able to handle large volumes of streaming and archival mobility data to infer trajectories. In particular, the focus is on maritime and aviation domains. It relies on Apache Flink and Kafka to handle large volumes of streaming data.

Aside from features already available on *scikit-mobility* [30], *moving-pandas* [12], *trackintel* [27], *ptrail* [14], *transBigData* [48], and *HuMobi* [38], SENDAS provides new functionalities on mobility analysis such as multiple shape AOI delineation and motif semantic annotation that none of these aforementioned solutions offer. Furthermore, SENDAS is built on top of Apache Spark which enables big data processing for all features. Note that *ptrail* has parallel processing by the *multiprocessing* package from Python, but this procedure is limited by the memory since the pool of processes does not use different worker nodes as in Apache Spark. Then, SENDAS is better compared to *ptrail* in this aspect. On the other hand, *TrajMesa* is a private system that is only accessed with credentials.

3 SENDAS

In this section, we describe SENDAS, a framework that provides methods to extract useful mobility patterns and metrics from large-scale datasets, based on the Apache Spark framework and the Apache Sedona library. In Section 3.1 we present how

Package Name	Languages	Features								
		Mobility Metrics	Flow Extraction	Stay Point Inference	AOI Inference	Home, Work Labeling	Multi-Shape AOI	Motif Extraction	Semantic Motifs	Big Data Processing
SENDAS	Scala	✗	✗	✗	✗	✗	✗	✗	✗	✗
TrajMesa	Scala, Python			✗						✗
SPARTAN	Java, Scala	✗		✗	✗					✗
scikit-mobility	Python	✗	✗	✗	✗	✗				
moving-pandas	Python	✗	✗	✗						
trackintel	Python	✗		✗	✗	✗				
ptrail	Python	✗								
transBigData	Python		✗	✗		✗				
HuMobi	Python	✗	✗		✗					

Table 1: Comparison of Mobility Frameworks

a trajectory can be represented in SENDAS, while in Section 3.2 we detail its mobility mining functionalities. Figure 1 presents the conceptual view and the main components of SENDAS, which are described in the following.

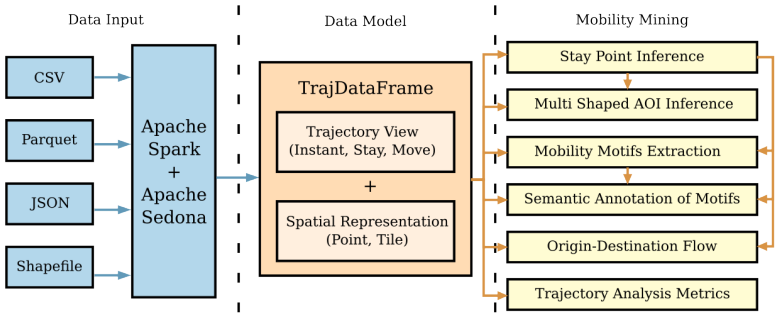


Fig. 1: Overview of SENDAS architecture.

3.1 Data Model

3.1.1 Trajectory Representation

SENDAS uses the following definition for trajectories as a base: a chronologically ordered sequence of points in space, belonging to a specific object [30, 52]. Since

this definition relies on a single temporal information, it is called an instant-based definition of trajectories.

Definition 1. *The instant-based trajectory of a moving object o is defined as a sequence of tuples $IT_o = \{(t_1, s_1), ..., (t_n, s_n)\}$, where s_i is a spatial object, t_i is a timestamp, and $t_i < t_j$ if, and only if, $i < j$.*

However, to capture other aspects of mobility semantics, two other trajectory definitions are also adopted. One of them comes from the concept of stay points [28], in which each data record represents the permanence of a moving object in a location for a given time.

Definition 2. *The stay-based trajectory of a moving object o is defined as a sequence of tuples $ST_o = \{(ts_1, te_1, s_1), ..., (ts_n, te_n, s_n)\}$, where s_i is a spatial object, ts_i and te_i are respectively the start and end timestamps of permanence in s_i, $ts_i < te_i$, and $te_i < ts_j$ if, and only if, $i < j$.*

The other definition is a line-based approach [12]. In this definition, we name data records as moves, since each of them represents the displacement of a moving object from an origin to a destination.

Definition 3. *The move-based trajectory of a moving object o is defined as a sequence of tuples $MT_o = \{(to_1, td_1, so_1, sd_1), ..., (to_n, td_n, so_n, sd_n)\}$, where so_i and sd_i are spatial objects respectively denoting origin and destination of the move, to_i is the departure timestamp from so_i, td_i is the arrival timestamp at sd_i, $to_i < td_i$, and $td_i < to_j$ if, and only if, $i < j$.*

These definitions denote complementary ways to represent trajectories and are thus called *trajectory views*. Also, the definitions are flexible to different *spatial representations*, i.e., what type of spatial data is used to represent the location of moving objects, to accommodate not only points in trajectory records, as described in Section 3.1.2. Thus, to fully characterize a set of trajectories, both the spatial representation and trajectory view should be specified when using SENDAS.

In Figure 1, a *TrajDataFrame* adds metadata to a regular Spark DataFrame in order to identify what kind of trajectory it represents. It can then be used as both input and output by other modules so that each trajectory type can be handled properly.

3.1.2 Spatial Representation

While mobility data is usually expressed in terms of points on a plane or on Earth, it is also useful to abstract each individual point into an aggregated notion of a region. For example, instead of indicating a person went from a given latitude and longitude to another, it is possible to indicate one went from city A to city B. For that purpose, we use the concept of tessellations to the spatial representation, which is the partitioning of space into geometries or tiles [21], and apply it in SENDAS as follows:

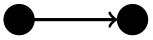
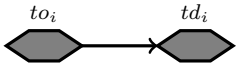
Definition 4. *A tessellation is a bijective function* $T : \Theta \to \Gamma$*, where* Θ *is a set of tile identifiers and* Γ *is a set of geometries.* $\forall g \in \Gamma$*,* g *supports containment operation with points* $p \in \mathbb{R}^n$*,* $n \geq 1$*.*

Tessellations can be feature-primary, dividing space according to real-world characteristics, such as districts of a city or country borders. They can also be space-primary, dividing space according to a function or an algorithm, independently of known features, e.g., Uber's Hexagonal Hierarchical Spatial Index H3 system [2]. Therefore, to support different types of tessellations in SENDAS, an implementation requires at least two functions, $T' : \mathbb{R}^n \to \Theta$ that, given a point $p \in \mathbb{R}^n$, returns the tile identifier θ of $g \in \Gamma$ where that point is contained, and the function T itself, returning the geometry g identified by θ.

Two tessellation functions are already implemented in SENDAS, the first being a user-defined tessellation, in which the user provides its own data set where each row is a tuple (θ_i, g_i). The function T' is then implemented using a spatial join query provided by Apache Sedona, that efficiently finds for every input point p_i which $g_j \in \Gamma$ satisfies $p_i \in g_j$, returning the associated θ_j. The function T is implemented with a regular join between the input set of identifiers and the stored set of identifiers Θ. The user-defined tessellation makes SENDAS flexible to be used for different purposes.

The second is the H3-based tessellation, derived directly from the H3 index [2]. It divides the surface of Earth into a set of pentagons and hexagons of varying resolution, from 0 to 15, where higher resolution values indicate smaller and more precise polygons, and each polygon is assigned a unique hash, forming its Θ set. H3 already provides the implementations of both T' and T, with the benefit that polygon data does not have to be stored beforehand since each identifier θ can be calculated solely from point coordinates.

Table 2: Trajectory Record Types

	Point	**Tile**
Instant	t_i 	t_i 
Stay	$[ts_i, te_i]$ 	$[ts_i, te_i]$ 
Move	to_i td_i 	to_i td_i 

Depending on what kind of spatial objects are used in a trajectory, we call it a point-based trajectory or a tile-based trajectory, as in Definitions 5 and 6, respectively. Table 2 summarizes all available combinations of trajectory views and spatial representations. The point-based trajectories, considered throughout this work, use the world geodetic coordinate reference system (CRS), with latitude (ϕ) and longitude

(λ) measured in degrees of arc in the respective ranges $[-90, +90]$ and $[-180, +180]$. However, SENDAS provides an interface that allows other coordinate systems to be implemented.

Definition 5. *A point-based trajectory is a trajectory whose spatial objects are points* $p_i \in \mathbb{R}^n$.

Definition 6. *Given a tessellation* $T : \Theta \rightarrow \Gamma$*, a tile-based trajectory is a trajectory whose spatial objects are tile identifiers* $\theta_i \in \Theta$.

3.2 Mobility Mining

To use SENDAS, first, the user has to indicate the trajectory and spatial representations that fit better to his/her interests. Next, the extraction of mobility patterns and metrics can be conducted, also according to the user's interests, as described in the following.

3.2.1 Stay Point Inference

When handling mobility data, one may be more interested in the stay points (i.e., locations where the user stays for a while) than in the raw locations. So, it is important to infer where are such stay points. In SENDAS, the process of finding the stay points involves transforming a point-instant trajectory into a point-stay trajectory based on [28]. It processes trajectory records linearly for each moving object, using three parameters to decide when to stop grouping consecutive points and start a new cluster: maximum distance d_{max}, minimum time t_{min}, and maximum time t_{max}.

Our proposed implementation is divided into two steps. First, a custom aggregator is used to enumerate to which cluster each instant point $IP_i = (t_i, p_i)$ belongs using the t_{max} and d_{max} constraints. The aggregator is mainly composed of a *reduce* function that takes as input an accumulator and a data frame row. In this case, the accumulator is a tuple $acc_i = (n_i, p'_i)$ and a row is an instant point IP_i along with the time difference $dt_i = t_i - t_{i-1}$, where n_i is the number of clusters processed until row i and p'_i is the starting point of the cluster containing row i. If either $dt_i > t_{max}$ or $distance(p'_i, p_i) > d_{max}$, n_i is incremented by 1 and p'_i is set to p_i to start a new cluster. For every row, the aggregator outputs n_i as its cluster index.

In the next step, the index n can then be used as a grouping key to calculate the stay points $SP_n = (ts_n, te_n, c_n)$, where $ts_n = \min_{\forall i, n_i = n}(t_i)$, $te_n = \max_{\forall i, n_i = n}(t_i)$, and $c_n = centroid(p_i | \forall i, n_i = n)$. The resulting stay points are then filtered out if $te_n - ts_n < t_{min}$. Figure 2 shows an example of the algorithm behavior. It should be noted that the contribution here involves the adaptation of the algorithm described in [28] to the large-scale nature of SENDAS.

3.2.2 Multi-shaped Area of Interest Inference

Different from a stay point, which consolidates multiple locations into a single one, an Area of Interest (AOI) is a region that a user visits frequently. The inference of

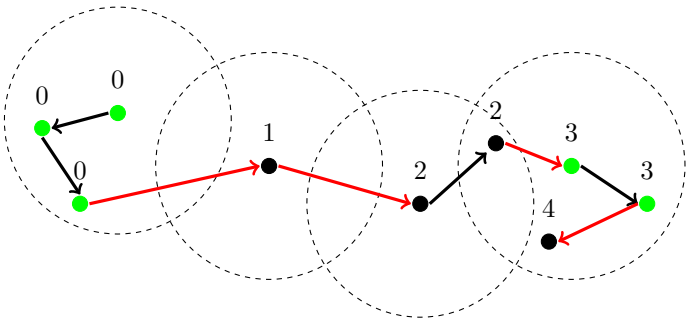


Fig. 2: Example of the stay detection algorithm steps. Labels indicate the cluster index n_i. Circles around starting points p'_i of each cluster indicate d_{max}. Red edges indicate either $dt_i > t_{max}$ or $distance(p'_i, p_i) > d_{max}$. Clusters 1, 2, and 4 would be discarded because their combined time difference is less than t_{min}, leaving clusters 0 and 3 as stay points, highlighted in green.

AOI of humans from mobility records is an important task to trace a user's preferences along the space. Since this problem provides meaningful insights for urban planning, transportation, communication networks, and the characterization of mobile user profiles [31], researchers have been putting effort into defining AOIs.

The authors of [28] calculate the stay region, a polygon that shapes a cluster of stay points, and compare the bounding box area resulting from the application of Grid-based clustering algorithm [50] and DBSCAN [9], a Density-based method. In [4], the authors improve the latter method to calculate points of interest (POI), including restrictions to temporal relation between the number of visits and the observable period of user trajectory. Nevertheless, it does not define the area of interest; the location of interest is characterized only by the centroid latitude and longitude. Differently from others that used GPS data to detect AOI, the authors of [16] used geotagged photos to extract locations. The method comprises a spatial-temporal layer that applies DBSCAN to cluster the photo locations, determining labels for each point. It delineates the AOI shape using the χ-Algorithm [7], to approximate the area more accurately. Besides this layer, it uses a semantic layer that defines a meaning for the AOI from the tags assigned to the photos collected from Flickr.

Given that an AOI is a region, it should be delineated into a shape. There are classical and novel methods to determine the polygon areas from a set of points. Examples of classical algorithms include the Convex Hull [11], the Minimum Convex Set ($\mathcal{CS}$) that contains all line segments $\overline{p_i p_j}$ formed by points $p_i, p_j \in S$, $i, j \in \{1, 2, \ldots, |S|\}$, and the Minimum Bounding Box (MBB), defined by the maximum and the minimum points on x-axis and y-axis. They are efficient methods to compute but contain empty areas.

To tackle this issue, algorithms based on concave hulls α-shape [8] and χ-Shape [7] were developed and can delineate the shape better than Convex Hull and MBB. However, these methods are optimized by parameters, and a novel algorithm [23] was developed to provide a model that does not consider any parameter to define the shape of a set of points. Nonetheless, while there are many methods to define the polygon

shape, works that model the AOI detection do not harness the full potential of those different approaches, which is an important contribution of SENDAS.

Considering all the mentioned studies, there is no work we are aware of that combines AOI detection, semantic assignment, and multiple options to shape delineation. Therefore, this module of SENDAS provides the processing pipeline, as Figure 3 shows, to infer and shape the area of interest from mobile users while supporting high volumes of data.

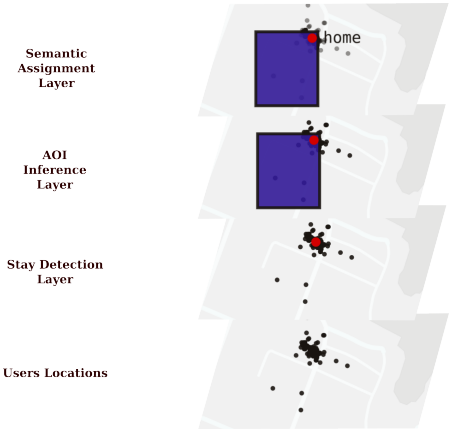


Fig. 3: AOI Inference Processing

The process to infer the AOIs A^u of user u can be defined formally as follows. Let $A^u = \{a_1^u, a_2^u, \ldots, a_n^u\}$, where a_i^u is a geometry representing one of the user AOIs. It is detected by clustering of stay-based trajectory ST^u, defined in Definition 2 in Section 3.1, where the moving object is a user and the spatial objects are the location points $l_i^u \in L^u$, where L^u is the trajectory ordered by a timestamp of l_i^u, represented by a tuple $l_i^u = \langle \text{latitude}, \text{longitude}, \text{timestamp} \rangle$.

Regarding the AOI detection method, the layers presented in Figure 3 are organized sequentially from the bottom-up perspective. First, the user stay points ST^u are extracted from the Stay Point Inference module discussed previously. After this, clustering is done by using the DBSCAN algorithm, which is efficient for this problem compared to other algorithms such as K-Means and is asymptotically acceptable $\mathcal{O}(n \log n)$, where n is the number of points.

The DBSCAN algorithm clusters such points given two parameters: *minPts* and ϵ_Ω, where Ω is the metric space. Then a cluster set cl_i should be stored if it contains at least *minPts* points where any of the points are distant no more than ϵ_Ω. These parameters are defined considering the objective of the problem; if the *minPts* is larger and if the trajectory L_u is somewhat sparse, there will be fewer clusters since the areas would not be filled up with minimum points to be considered as clusters. Similarly, if the ϵ_Ω is smaller, it will add a hard constraint, as the points ought to be close to each other to be considered clusters. Accordingly, SENDAS provides an optional argument to set the metric to use, which could be a Haversine distance, Euclidean Distance or

Manhattan, and also, the *minPts* and ϵ_Ω DBSCAN parameters to define the cluster set of points $CL^u = \{CL_1^u, CL_2^u, \dots CL_n^u\}$.

After discovering a cluster set of points CL_i^u using the DBSCAN, the area a_i^u should be delineated in order to characterize a polygon $P \subset \mathbb{R}^2$ that contains the cluster. Considering that, in mathematics, there are only three types of polygons $P \subset \mathbb{R}^2$: regular, convex, and concave. SENDAS provides the option to delineate P using a Minimum Bounding Rectangle, which is a regular polygon, using a Convex Hull from Graham's Scan, which defines a convex polygon, and lastly using a Concave Hull defined from a Delaunay Triangulation. Then the module provides a multiple-shape alternative to define the AOI a_i^u given a cluster CL_i^u, distinguishing from other methods that shape the AOI with only one type of polygon.

In addition to inferring the AOI, in SENDAS it is also possible to assign its semantic information. As the final layer of the sequence, the semantic assignment is based on the definition of the most visited AOI in a timestamp [4]. Each AOI is classified into three classes: Home, Work, and Other. Any $a_i^u \in A^u$ not assigned as Home or Work is automatically assigned as Other.

The AOI $a_i^u \in A^u$ is defined as Home if it is the most visited area in the evening-morning (8:00pm to 8:00am) by u, that is, it contains the most number of l_i^u compared to others AOIs $A^u \setminus a_i^u$. Note that there is only one AOI classified as Home. This can be formulated as a function $Home(A^u) = argmax_{a_i^u} \frac{n_i}{|L_{ct}^u|}$, where n_i is the number of visits within the a_i^u subjected to the time constraints, the L_{ct}^u is the set of location points constrained by time, in this case, the aforementioned evening-morning time interval.

To classify an AOI as Work, is considered the set $A^u \setminus Home(A^u)$, to simplify, this set will be considered as $A_{H^-}^u$, which represents the set of AOI without the one classified as Home. Hence, the function $Work(A_{H^-}^u) = argmax_{a_i^u} \frac{n_i}{|L_{ct}^u|}$ classifies A_i^u as a Work location if it is the most visited area in morning-evening (9:00am-7:00pm). Hence, the elements $A_{W^-}^u$ that are left in the set of AOIs are classified as Other locations. This method does not consider the inactive interval defined by [4], since this adds computational complexity in handling such a high volume of data. They define an inactive interval after inspecting all records of a user u with n records, which results in a complexity of $\mathcal{O}(un)$, thus if there are several users with many records, this process would last quite long. On that account, the module of AOI inference has been developed assuming a fixed time interval.

3.2.3 Mobility Motifs Extraction

To capture repeating patterns of individual moving objects, we use the concept of mobility motifs [36]. In analogy to complex networks, if movements are organized into a temporal network, – i.e., a temporal graph – a mobility motif is a recurring subnetwork that represents the visited locations of a moving object along a time slot. Since no moving object can ever be in two places simultaneously, it is reasonable to assume a strict order among the temporal edges, unlike other works, related to temporal motifs, which address possibly simultaneous events [20, 22, 41]. While the definition of temporal motifs given by these works applies to ours, it is a more general approach that handles the case of simultaneous temporal edges.

Furthermore, following the three-step motif identification process proposed in [20] – (1) find all maximal connected subgraphs E^*_{max}, (2) find all valid subgraphs $E^* \subseteq E^*_{max}$ and (3) identify the motif corresponding to E^* –, it is possible to simplify those steps by using the more restricted mobility motif definition as follows to make certain assumptions. This, along with the parallelized implementation, is another important contribution of SENDAS, which allows the extraction of mobility motifs from large-scale datasets.

Definition 7. *The motif M_{ot} of a moving object o during time slot $t = (t_{start}, t_{end})$ is a temporal graph $M_{ot} = (V, E)$, where V is the set of visited regions and E is the set of strictly ordered temporal edges $E = \{(u_i, v_i, t_i) | u_i, v_i \in V, t_{start} \leq t_i < t_{end}\}$.*

The first step, as aforementioned, is accomplished by dividing movements into time slots. As for the second step, since we are concerned with capturing the mobility pattern of each object, and not patterns on the whole network formed by all movements, it is enough to assume that the entire sequence of moves performed by that object comprises its motif, for a given time slot. Finally, to count how many times each motif was detected for each object, or appeared in the entire data set, it is necessary to compare those graphs and check if they have the same structure, i.e., if they are isomorphic.

General subgraph isomorphism is an NP-complete problem, but if restrictions are imposed in their structure, polynomial algorithms can be devised [17]. Using our definition, we propose an $O(E \log E)$ technique to assign canonical labels to motifs and use those labels to identify isomorphic motifs, where E is the number of motif edges.

As seen in Figure 4, each vertex corresponds to a region, which can be an AOI or a tile, visited by a moving object, and the temporal order of edges is indicated by their labels. Let $E_T = \{e_i | e_i = (u_i, v_i)\}$ be the temporally ordered sequence of edges of a temporal graph $G_T = (V, E)$, and let $\Sigma = \{\sigma_0, \sigma_1, ... | \forall \sigma_i, \sigma_j, i < j \rightarrow \sigma_i < \sigma_j\}$ be a set of unique labels, we can define a bijective function $\sigma : V \rightarrow \Sigma$ that maps each vertex of a temporal graph to a unique label from Σ, according to the order of their first appearance in E_T, i.e., the first vertex in an edge of E_T is mapped to σ_0, the second is mapped to σ_1, and so on. Formally, let $e_0 = (u_0, v_0)$ be the first edge of E_T, then $\sigma(u_0) = \sigma_0$. Also $\forall e = (u, v) \in E_T, \sigma(u) \leq \sigma(v), u = v \leftrightarrow \sigma(u) = \sigma(v)$ and $\forall e_i, e_j \in E_T, i < j \rightarrow \sigma(v_i) \leq \sigma(u_j), v_i = u_j \leftrightarrow \sigma(v_i) = \sigma(u_j)$.

The canonical label of a graph G is defined as $\tau^*(G) = \{(\sigma(u_i), \sigma(v_i)) | 0 \leq i \leq E\}$, mapping each ordered edge of E_T with σ, but a proof is necessary to ensure $\tau^*(G)$

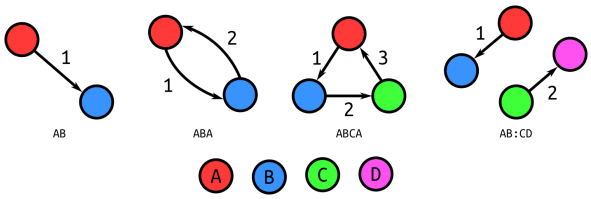


Fig. 4: Examples of mobility motifs and their assigned canonical labels.

is indeed a canonical label. Considering that if, and only if, temporal graphs $G_A = (V_A, E_A)$ and $G_B = (V_B, E_B)$ are isomorphic, there is a function $\varphi : E_A \to E_B$, we must show by transitivity that $\exists\varphi : E_A \to E_B \leftrightarrow \tau^*(G_A) = \tau^*(G_B)$, i.e, there is a bijective function φ that maps every edge of G_A to an edge in G_B preserving the temporal order if, and only if, the canonical labels are equal.

First, we prove that if such function φ exists, then the canonical labels given by τ^* must be equal. The first condition of φ states that every edge in E_A must be uniquely mapped to an edge in E_B, which means that $|\tau^*(G_A)| = |\tau^*(G_B)|$. The second condition states that the temporal order of the edges is preserved after the mapping. Thus, the first vertex that appears on $E_{T,A}$ is mapped to the same label in Σ as the first vertex that appears in $E_{T,B}$, and the same applies to every subsequent vertices. Hence, $\tau^*(G_A) = \tau^*(G_B)$.

To aid in the second half of the proof, we can define a bijective function $\tau : E \to \Sigma \times \Sigma \times \mathbb{N}$ that maps each edge of the graph G to a tuple $\tau(e) = (\sigma(u), \sigma(v), i)$, where i is the index of e in the sequence E_T or $\tau^*(G)$.

Next, we prove that if $\tau^*(G_A) = \tau^*(G_B)$, then it is possible to construct a bijective function φ that meets the temporal isomorphism criteria. Such function can be simply defined as $\varphi(e_A) = \tau'_B(\tau_A(e_A))$, in which $\tau_A : E_A \to \Sigma \times \Sigma \times \mathbb{N}$ maps edges in E_A to their vertices labels and their order in $E_{T,A}$, and $\tau'_B : \Sigma \times \Sigma \times \mathbb{N} \to E_B$ is the inverse of τ_B and maps vertex labels and order back into edges of E_B. Having τ^* defined for both graphs implies that τ_A and τ'_B can be defined. Since $\tau^*(G_A) = \tau^*(G_B)$, their length must also be equal, which means every edge in E_A can be mapped to an edge in E_B. Moreover, since the order is also the same and is unambiguous, it is guaranteed that for any two edges e_{Ai} and e_{Aj} in E_A, if e_{Ai} appears before e_{Aj} in the canonical label, there are two corresponding edges e_{Bi} and e_{Bj} in E_B given by φ that also respect that order. Hence, the proof is complete.

As seen in Figure 4, the chosen Σ set is the standard upper case alphabet letters in ASCII. The canonical label $\tau^*(G)$ is transformed to a string by simply concatenating each $\sigma(u)$, with the following rule. For each consecutive $(\sigma(u_i), \sigma(v_i)), (\sigma(u_{i+1}), \sigma(v_{i+1})) \in \tau^*(G)$, if $\sigma(v_i) = \sigma(u_{i+1})$, $\sigma(u_{i+1})$ is omitted, or else, a break character is inserted before it – a colon character was arbitrarily chosen as the break –, as shown by the `AB:CD` motif. Finding $\tau^*(G)$ and transforming it to a string takes linear time, i.e., it is $O(E)$, but the edges must be previously sorted, which is commonly an $O(E \log E)$ operation, thus the total complexity is $O(E \log E)$.

The motif `AB:CD` represents a moving object o trajectory $\{A, B\} \bigcup \{C, D\}$, which means a discontinuous trajectory because of data collection or data sparsity. It is reasonable to assume an edge from B to C is missing since moving objects can not jump between locations without traveling the space in between. Therefore, SENDAS provides the user two options: show the break character to enforce the trajectory discontinuity or to omit every break character, as Figure 5 shows.

The final output for this feature consists of every moving object identifier associated with each time slot and their corresponding motif canonical label.

After outlining these definitions and proving our proposal, we further improved state-of-the-art motif labeling techniques in temporal graphs by limiting the scope of the problem to the characteristics of mobility data. Such findings contribute to

research and applications in large-scale movement pattern analysis by providing an explicit model to identify mobility motifs, independently of extraction methods.

3.2.4 Semantic Annotation of Motifs

While mobility motifs allow researchers to model mobility patterns [36], they do not characterize the daily trajectories of mobile users semantically. Then, since it is not possible to study the preferences over locations with regular motifs, there are many options to assign semantic meaning to them.

One alternative is to use data from surveys and the government. The authors of [40] use the daily mobility motif and trip purposes from California of the 2017 National Household Travel Survey (NHTS) to support their correlation analysis between senior people's characteristic displacements and workdays and non-workdays. The survey collects the origin and destination of a trip, which can be a home, a workplace, or other trip purpose. Therefore, it is possible to build daily motif networks representing the displacement. In [39], the same survey (NHTS) was used to assign the semantics to the motifs alongside the demographic data to show the differences between commuters' and telecommuters' routines. Considering the North American Classification System (NAICS), the authors of [26] labeled all POI data collected from Safe Graph, and builds the motifs with semantics related to the visits to the POI data. In addition, they assign the characteristic place categories that are the most visited ones according to the different types of motifs.

Another alternative is to infer the semantic meaning of the vertex representing the motif network through trajectory data mining methods. In [3], the authors use data from mobile phone positioning in Shenzhen city of 9.702.082 users and extracted two types of motifs: the location-based motif, which is similar to the unannotated mobility motif, and the activity-based motif which is the semantic motif. To construct the semantic motif, the authors detect three types of activities – work, home, and social – according to the data event timestamp, using the circadian rhythm to infer the activity. In the work [42], the authors use the same dataset of Shenzhen city to mine the spatial relations through the motif patterns. The authors of [45] use Wuhan Call Detail Records (CDRs) to analyze users' mobility during the day by mining motif patterns. To infer the semantic meaning of an activity, they proposed the detection of anchor points, that define the weighted center of a grid cell as the location of the user in a timestamp. Thereafter, the inference of activity type, i.e., Home, Work, and Other, is detected using the circadian rhythm as in [3]. Then, they build the motif with the activity semantics.

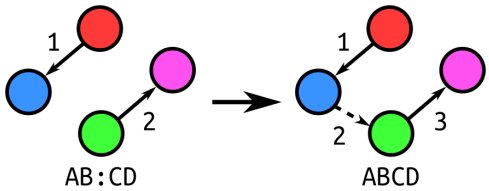


Fig. 5: Discontinuous mobility motif and its continuous version.

Therefore, it is noticeable that the state-of-art of semantic motif inference through mobile data phone records uses only 3 classes: Home, Work, and Other due to the difficulty of relating the category data from POI extracted from Volunteered Geographic Information Databases with stay points of mobile users [32]. Moreover, all aforementioned works lack a centralized method to be reproducible for other researchers; they present differences that hinder the decision of what method to follow. Finally, their solutions are not adequate for large-scale datasets. Therefore, the method proposed here and implemented in SENDAS aims to centralize the process of motif semantic inference with nodes classified as Home, Work, or Other location for large-scale datasets. This is another relevant contribution of our work.

In Section 3.2.2 we detailed the inference of semantic area of interest into Home, Work, or Other using the visits frequency discretized by the time-interval, i.e., evening-morning, morning-afternoon, afternoon-evening. Subsequently, each first letter of semantic class $\sigma_C = \{H, W, O\}$ is assigned to nodes in motif network, that is, H for Home, W for Work, and O for Other. Hence, the semantic motif can be defined as follows. Let E_T be the edges of a temporal graph $G_T = (V, E)$, and let Σ_C be a set of semantic labels, we can define a bijective function $\sigma_c : V \to \Sigma_C$ that maps each vertex V to a label from Σ_C, according to the AOI semantic class a_i^u visited. Thus, let an edge $e(u_i, v_j) \in E_T$, the u_i and v_j are labeled with the class of a_i^u and a_j^u through the function σ_c, i.e., if the a_i^u is classified as home, the vertex u_i will be labeled as H, and v_j will be labeled as W if the a_j^u is classified as Work, then the edge $e(u_i, v_j)$ represents the displacement from Home to Work. Therefore, the semantic label of a graph G is defined as a list of tuples $\tau_c^*(G) = \{(\sigma_c(u_0), \sigma_c(v_0)), \ldots, (\sigma_C(u_E), \sigma_c(v_E))\}$ mapping each edge of E_T with σ_c, the proof of 3.2.3 also applies here.

The motif semantic inference method requires a move-based trajectory, as stated in Definition 3, where the spatial objects so_i and so_j are the semantic labels. Therefore, SENDAS can provide all steps to infer the semantic motif by providing a point-based trajectory: (1) extract stay points, (2) detect the AOI, (3) classify the AOI, (4) generate the semantic move-based trajectory, (5) infer the semantic motif. Hence, it centralized all the methods to build a complete trajectory mining pipeline, providing an environment to analyze human mobility patterns.

3.2.5 Origin-Destination Flow

Origin-destination flows are used as an aggregated metric for mobility analysis, which measures the number of movements between pairs of regions. It is often useful to analyze flow along time, so we extend it with a temporal dimension, splitting flow into *inflow* and *outflow*, as stated in Definition 8.

Definition 8. *Flow is defined as a set of matrices Φ. $\forall F_t \in \Phi$, F_t is an $O \times D$ matrix, and $F_{t,OD} = (f_{t,od}^{out}, f_{t,od}^{in})$, where $f_{t,od}^{out}$ is the number of moving objects departing from region θ_o at time slot t towards region θ_d, $f_{t,od}^{in}$ is the number of moving objects arriving at region θ_d at time slot t, coming from region θ_o, and O, D are the number of distinct origin and destination regions, respectively.*

Flows are calculated from tile-move trajectories, because this type of trajectory simultaneously represents movements explicitly, and the locations are represented by tiles instead of points, enabling them to be used as grouping keys. From Definition 3, flow metrics $f^{out}_{t,od}$ and $f^{in}_{t,od}$ are therefore calculated as

$$f^{out}_{t,od} = |\{(to_i, td_i, so_i, sd_i) | \forall i, to_i \in t, so_i = \theta_o, sd_i = \theta_d\}|$$

$$f^{in}_{t,od} = |\{(to_i, td_i, so_i, sd_i) | \forall i, td_i \in t, so_i = \theta_o, sd_i = \theta_d\}|.$$

The time slots t can be defined as uniformly distributed ranges of time, such as hours or days. It is also possible to specify the offset after which the time slots begin, e.g., for time slots representing a social day, each slot should last 24 hours and start at 5 PM, instead of 12 AM.

SENDAS allows further categorization of time, such as labeling the hours of the day as *morning*, *afternoon* or *night*, or labeling days of the week as *weekday* or *weekend*. In this case, the conditions $to_i \in t$ and $td_i \in t$ also take into account if the categories of the timestamps to_i or td_i match the categories of time slot t. If no division of time slots or time categorization is used, only a single time slot will exist, encompassing the whole set of trajectories. Consequently, by definition $f^{out}_{t,od} = f^{in}_{t,od}$ for any pair of regions.

Some tools, such as *scikit-mobility* [30], offer a feature to extract flow. However, the parallelized implementation of SENDAS allows the extraction of flows from large-scale datasets and is another contribution of this work.

3.2.6 Trajectory Analysis Metrics

Exploratory data analysis, followed by filtering, clustering, and other transformations are usual practices in data science in general, and mobility analysis is no exception. Thus, SENDAS provides functions for introductory extraction of trajectory metrics [12] (Table 3), and a set of algorithms necessary for the main use cases presented in this work. It is important to mention that the contribution of SENDAS relies on big data processing to extract such metrics.

Duration, distance, and velocity are calculated for each point present in the trajectory, while gyradius, k-gyradius, and uncorrelated entropy are calculated for each moving object o. Also, distance, velocity, gyradius, and k-gyradius only apply to point-based trajectories. Using Definitions 1, 2, 3 and 5, each metric is defined as follows.

Duration is calculated differently depending on the trajectory view: for instant-based trajectories, it is the difference $t_{i+1} - t_i$, for stay-based trajectories, $ts_{i+1} - te_i$, and for move-based trajectories, $td_i - to_i$.

Distance is calculated as the haversine distance $d(s_i, s_{i+1})$ for both instant and stay-based trajectories, and $d(so_i, sd_i)$ for move-based trajectories, as defined by Equation 1, where $R = 6.371 \times 10^6$ is the Earth average radius in meters.

$$d(a, b) = 2R \arcsin \sqrt{\sin^2\left(\frac{\phi_b - \phi_a}{2}\right) + \cos\phi_a \cdot \cos\phi_b \cdot \sin^2\left(\frac{\lambda_b - \lambda_a}{2}\right)} \quad (1)$$

Table 3: Metrics available in SENDAS.

Metric	Description
duration	Calculates the time taken to move between consecutive trajectory points.
distance	Calculates the distance between consecutive trajectory points.
velocity	Calculates the average speed between consecutive trajectory points, along with the angle formed by the points.
gyradius	Aggregates every point of each moving object to calculate its radius of gyration.
k-gyradius	Aggregates the k most visited points of each moving object to calculate its radius of gyration.
uncorrelated entropy	Computes the probability of a moving object returning to a location already visited.

Velocity is a vector formed by an origin and a destination point with a speed and angle component. The speed is calculated as distance divided by duration, depending on the trajectory view. The angle is the absolute bearing, i.e., the angle between the north pole and the vector, as defined by Equation 2. For instant- and stay-based trajectories, it is $\alpha(s_i, s_{i+1})$, and for move-based trajectories, it is $\alpha(so_i, sd_i)$.

$$\begin{aligned}\alpha(a,b) = \operatorname{arctan2}(&\cos\phi_b \cdot \sin(\lambda_b - \lambda_a), \cos\phi_a \cdot \sin\phi_b \\ &- \sin\phi_a \cdot \cos\phi_b \cdot \cos(\lambda_b - \lambda_a))\end{aligned} \tag{2}$$

To obtain the gyradius, or radius of gyration, it is necessary to calculate the centroid c_o from all points s_i of a moving object o, according to Equation 3, where n is the number of points. Finally, the radius of gyration rg_o is set to the average distance between every point to the centroid (Eq. 4).

$$\begin{aligned} c_o(S) &= \left(\arcsin\frac{z}{n}, \operatorname{arctan2}(y,x)\right) \\ x &= \sum_{s_i \in S} (\cos\lambda_i \cdot \cos\phi_i) \\ y &= \sum_{s_i \in S} (\sin\lambda_i \cdot \cos\phi_i) \\ z &= \sum_{s_i \in S} \sin\phi_i \end{aligned} \tag{3}$$

$$rg_o = \frac{1}{n} \sum_{s_i \in S} d(c_o, s_i) \tag{4}$$

Similarly, to compute the k-radius of gyration, it is necessary to calculate the centroid c_o^k using the k locations most visited represented in Equation 5 as a set S^k.

$$rg_o^{(k)} = \sqrt{\frac{1}{|S^k|}\sum_{i=1}^{k} d(rg_o - c_o^{(k)})^2} \tag{5}$$

The difference between the radius of gyration and the k-radius of gyration is the k, which is a parameter that determines the k most frequently visited locations of a moving object o, whilst, the rg_o represents all locations of a moving object o.

The uncorrelated entropy estimates the routine predictability of a moving object o. If the entropy is high, the moving object visits different locations, while the low entropy represents a predictable routine.

$$E_{unc}(o) = -\frac{\sum_{j=1}^{N_o} p_o(j) log_2 p_o(j)}{N_o} \tag{6}$$

In equation 6, the N_o is the set of distinct locations of the moving object o, and the $p_o(j)$ is the probability of o has already visited the location j.

All aforementioned metrics can be extracted from large-scale datasets in a parallelized manner. The user can then use the results to extract characteristics from the mobile users contained in the dataset.

4 Performance Evaluation

To assess the scalability and efficiency of *SENDAS*, three scenarios are proposed to illustrate its currently implemented main features: stay detection, flow calculation, and motif labeling. Then, comparative experiments were conducted to evaluate how the number of parallel execution threads used by Spark can leverage performance.

The GeoLife dataset [51] (version 1.3) was used as input for the experiment. It is a GPS trajectory dataset collected from 182 users from April 2007 to August 2012, with varying sampling rates. It is a point-instant-based trajectory dataset since each data record is composed of a user identifier, a timestamp, and latitude and longitude coordinates. There are $24,876,978$ records in total, which is suitable for a large-scale analysis.

All tests were performed on a cluster, with the following specifications: 2 AMD Opteron 6376 (16M Cache, 2.3 GHz, 32 cores) processors with 512 GB RAM. For each test instance, the corresponding relevant code section was run 10 times and the total running time was measured, to avoid disparities due to other processes running on the same machine. Furthermore, the performance of SENDAS is measured with different amounts of execution threads to evaluate if parallelism is properly contributing to reducing total running time.

The first experiment is with the stay detection algorithm, whose output is later used by the next experiments. To provide a baseline for the time measures, the algorithm proposed by [28] was implemented directly in Python. The algorithm parameters were chosen empirically as $t_{min} = 10$ min, $t_{max} = 12$ h and $d_{max} = 200$ m. In later

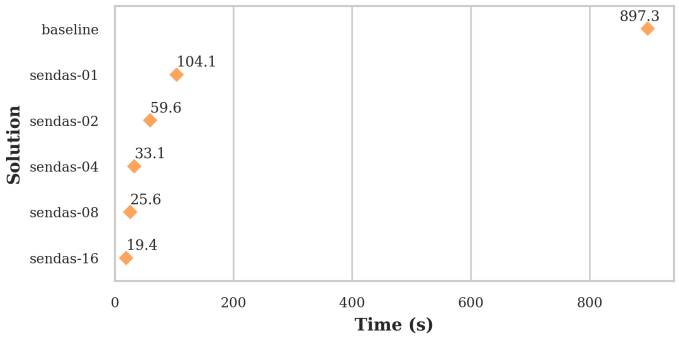


Fig. 6: Running time for stay detection in seconds.

experiments, the H3 tessellation would be used with resolution 9, therefore d_{max} was specifically set to 200 m to match the average edge length of polygons in that resolution.

In both the baseline and SENDAS solutions, the measured execution time considers reading input files, running the stay detection algorithm, and writing the output to disk. Figure 6 shows the test results, with the average running time and the corresponding 95% confidence interval. The number suffix on each SENDAS solution refers to the number of execution threads used. The difference between the baseline solution and the single thread SENDAS solution is already meaningful, with SENDAS running approximately 8.6x faster. And, as expected, execution time is decreased as the number of threads doubles, with up to 1.8x improvements from 1 to 2, then 4 threads, and less expressive improvements of up to 1.3x from 4 to 8, then 16 threads. Comparing the single thread solution to the 16 threads solution, there is a 5.3x improvement in running time.

A total of $27,141$ stay points were detected, with an average number of stays per user of 151.63, and an average 3.12 daily stays per user. The resulting stay points were then used as input for the next experiment – flow calculation. Due to the reduced number of records compared to the original dataset, the stay trajectories were oversampled to maintain a similar amount of data across all experiments. In total, 100 copies of the stay points were made, with unique user identifiers, leading to $2,714,100$ rows.

In the next experiment, we used *scikit-mobility* [30] implementation as a baseline, which requires a user-defined tessellation to assign tile identifiers to data records. To make a fair comparison between SENDAS and *scikit-mobility* flow calculation, the H3 tessellation is used with resolution 9 to obtain the polygons present in the trajectories. The polygons and their identifiers were then exported to be used as a user-defined tessellation, with a total of $5,697$ unique polygons. This way both libraries need to perform the same spatial join queries. Moreover, since *scikit-mobility* does not use the same concept of time slots to calculate flow, no time division or categorization is used.

After preparing the input data, the experiment measures for *scikit-mobility* the time to read input files, assign tile identifiers, run the flow calculation algorithm, and

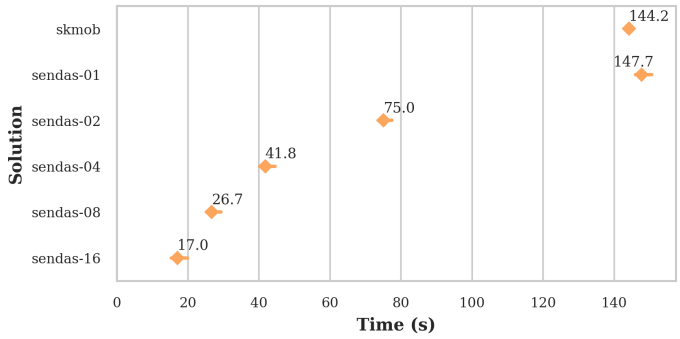


Fig. 7: Running time for flow calculation in seconds.

write the output to disk. For SENDAS, it is necessary to convert the tile-stay trajectories to tile-move trajectories before the flow calculation, as mentioned in Section 3.2.5, which can be achieved with the following transformation present in the library. Given a stay-based trajectory ST_o (Def. 2), a move-based trajectory MT_o (Def. 3) can be derived as $MT_o = \{(te_i, ts_{i+1}, s_i, s_{i+1}) | \forall 1 \leq i < n\}$.

The experiment results are shown in Figure 7, with a total of 16,211 unique origin-destination pairs calculated. It is possible to see the *scikit-mobility* solution, labeled as *skmob*, and SENDAS single thread solution with similar execution times, differing only by 3 seconds. That can be explained by the fact that SENDAS needs an additional transformation step on the input, and by how flow is calculated both for outflow and inflow, even when no time division is used. However, as the number of threads increases, execution time gets shorter by a factor from 1.9x to 1.5x. In total, there is a 8.7x improvement from the single-thread solution to the 16-thread solution.

Finally, for the motif extraction experiment, we use the H3 tessellation directly instead of using the stored tessellation, performing the following steps. Trajectory data is loaded into memory, then transformed to the tile-stay format, and later to tile-move, using the aforementioned technique. The motif labeling algorithm is executed to identify daily motifs, which are then grouped for each user, counted, and sorted from the most frequent to the least. The final result is written to disk. A total of 537 unique motifs were identified, with 12.1 unique motifs per user on average.

Once again, Figure 8 shows an inversely proportional relationship between the level of parallelism and execution time. While a single thread takes around 27 seconds to perform its task, 16 threads take around 5 seconds to finish, an overall improvement of 5.7x. This demonstrates the efficiency of SENDAS, which was designed and implemented following Big Data techniques, in comparison to single-threaded traditional solutions.

5 Case Study

This section presents a mobility analysis during and after the COVID-19 pandemic, specifically comparing human displacement patterns between 2021 and 2022 in Brazil. Through the SENDAS execution, mobility signatures were extracted using semantic

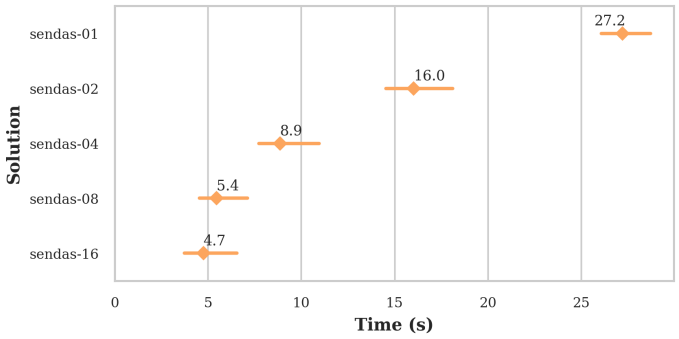


Fig. 8: Running time for motif labeling in seconds.

motifs and metrics, revealing how human patterns evolved during this period. The main goal here is to evaluate the applicability of the proposed framework to real-world problems using a large-scale dataset. Furthermore, the secondary contribution is to outline how mobility patterns evolved throughout the pandemic period.

5.1 Data and Methods

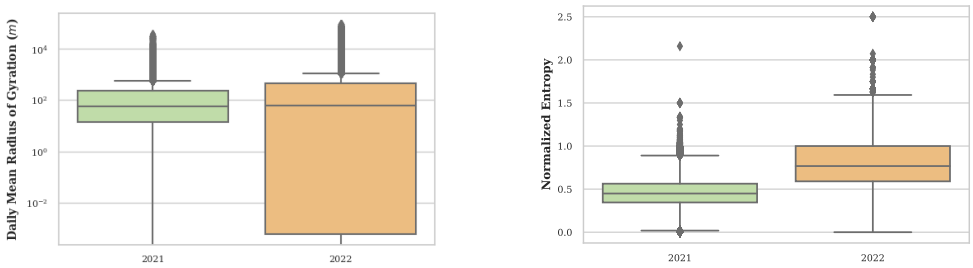
It was used a real-world dataset, made available by a private company, with 49.435.713 records in 2021 and 46.087.099 records in 2022, from a total of 4.279.025 anonymous users.

To characterize the mobility signatures of each user, it is necessary to apply the methods and compute different metrics as follows: (1) detect the stay points from events; (2) discover AOIs from stay points; (3) assign the semantic (Home, Work and Other) to AOIs; (4) extract the mobility signatures by mining motifs from displacements of semantic AOIs; and (5) calculate mobility metrics and the motif transition matrix.

To identify the stays points of each user, it was used the stay detection module of SENDAS with D_{max}, T_{min} and T_{max} equal to 200 meters, 10 minutes, and 8 hours, respectively. D_{max} assumed this value since is the lowest value considered which produces good results [28]. T_{max} was defined as 8 hours because, in general, people stay at work for at most this time. T_{min} was set to 10 minutes to ignore stays in traffic due to congestion or traffic lights.

After detecting the stay points, the AOIs were calculated using the AOI Detection module of SENDAS with the same value for parameters of DBSCAN [4], thus $epsilon_{\Omega}$ is equal to 100 meters and the minimum of samples is equal to 0.18 of the dataset length. In addition to that, the metric space is defined in $\mathbb{R}^3$ set where the Haversine was considered as distance function.

Hence, with the AOIs calculated, it is possible to assign semantic information to them using AOI inference. This step is the fundamental one to produce the mobility signatures, since the comparison displacements can be viewed as the displacements to the places classified as Other in our work, e.g., malls, restaurants, and food courts.



(a) Radius of gyration. (b) Uncorrelated entropy.

Fig. 9: Mean radius of gyration distribution and Uncorrelated entropy distribution.

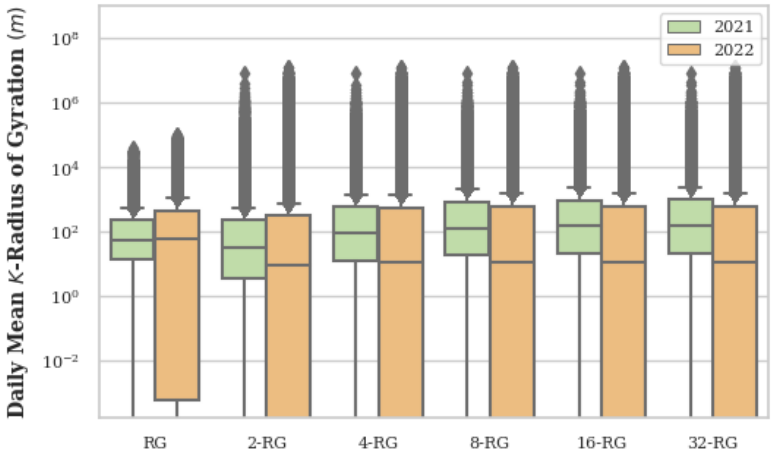


Fig. 10: K-radius of gyration distribution of periods.

For example, if a user visits more Other locations than Home and Work, this person can be classified as explorer [29].

The execution was conducted in a machine equipped with Intel i7-1165G7 (12M Cache, 2.8 GHz, 8 cores, 2 threads) processor, 16 GB DDR4 3200 MHz RAM and 512 GB of hard disk. It took around 2 hours to process all the data and extract the results.

5.2 Mobility Metrics

Figure 9a presents the mean radius of gyration distribution per user. It is observable that, although the medians are close, in 2022 the variance is significantly bigger. This can be explained by users visiting more places next to their center of mass in 2022, decreasing their radius of gyration.

Figure 9b presents the user's entropy, showing a less predictability in individuals' displacements in 2022 compared to the previous year because of the higher entropy.

Year	k	Returners	Explorers
2021	2	47,79%	52,21%
2021	4	67,78%	32,22%
2021	8	73,30%	26,70%
2021	16	74,24%	25,76%
2021	32	74,31%	25,69%
2022	2	32,59%	67,41%
2022	4	40,91%	59,09%
2022	8	41,99%	58,01%
2022	16	42,05%	57,95%
2022	32	42,06%	57,94%

Table 4: Classification of Returners and Explorers

Thus, users present more variability in their daily trajectories, making human mobility patterns harder to predict.

Finally, Figure 10 shows that the k-radius of gyration is lower in 2022 than in 2021. As more places are visited, the k places more frequently tend to have a lower relevance in the contribution to the radius of gyration. For a complementary mobility study, Table 4 shows the users' classification in Returners and Explorers following the definition of [29]. According to this categorization, in 2021 the proportion of returners is bigger than explorers, which reveals the tendency of users to shift to known places. On the other hand, the tendency in 2022 is the opposite, while there are more explorers than returners since users began seeking novel places after the pandemic's restrictions eased.

5.3 Semantic Motif

In this analysis, we assume 'W', 'H', and 'O' as Work, Home, and Other, respectively. Figure 11 shows some changes in mobility patterns when comparing the semantic motifs. In 2021, the motifs present a few displacements since they have a short length, such as 'WO'. Other frequent motifs are predominantly characterized by Home-Work (HW) or Work-Home (WH) commuting. Conversely, in 2022, it is observable a wide-ranging of motifs, with longer sequences and visits to other places than home and work, including 'OHO', 'HWO', 'WOH'. Besides that, it is noted an influence by the year in motifs patterns as the 'WOW' sequence occurs frequently in 2022 while in 2021 was uncommon since shifting to a restaurant, for example, during work routine was unlikely due to COVID-19 mobility restriction.

6 Conclusions and future directions

In this work, we proposed SENDAS, a new Scala library built on top of Apache Spark and Apache Sedona. The technical contribution of our work is a scalable and flexible library to process mobility datasets and extract useful knowledge from them. We assess the performance of SENDAS running for different levels of parallelism, finding improvements up to 8.6 times when compared to non-parallel solutions.

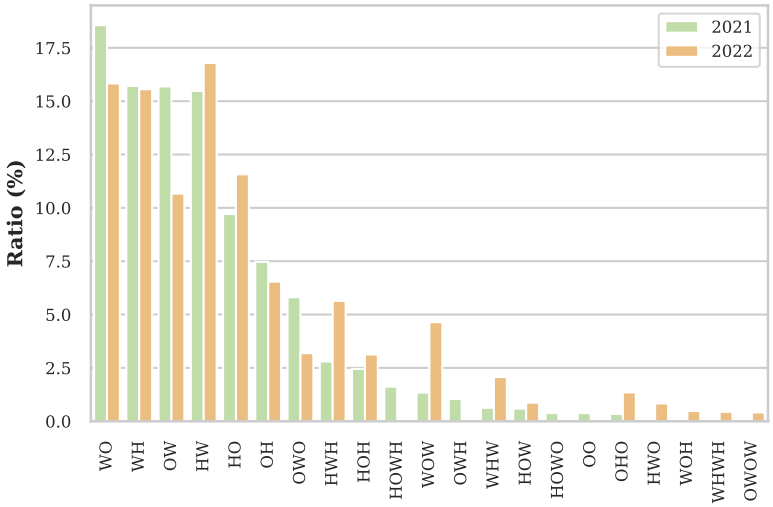


Fig. 11: Semantic Motifs Frequency

The main scientific contributions of this work are three-fold. First, as for motif extraction, we defined more strictly what motifs mean in the mobility context, enabling an asymptotically less complex method for temporal graph isomorphism, and defined the motif semantic annotation. Second, we add a temporal dimension to flow calculation to make the results more flexible. Finally, we propose a multi-shape area of interest delineation. These contributions are relevant to the research community in the area of human mobility and communications. It should be noted that all these contributions also rely on the big data nature of SENDAS.

Future developments in the library include expanding the available analysis and transformation methods to match those of similar packages like *scikit-mobility* and *MovingPandas*. It is also relevant to add plotting features for Scala practitioners since many visualization tools have been implemented and adopted for Python and R.

Declarations

No funding was obtained for this study.

References

[1] Barbosa H, Barthelemy M, Ghoshal G, et al (2018) Human mobility: Models and applications. Physics Reports 734:1–74. https://doi.org/https://doi.org/10.1016/j.physrep.2018.01.001

[2] Brodsky I (2018) H3: Uber's Hexagonal Hierarchical Spatial Index. URL https://github.com/uber/h3

[3] Cao J, Li Q, Tu W, et al (2019) Characterizing preferred motif choices and distance impacts. Plos one 14(4):e0215242

[4] Capanema CGS, Silva FA, Silva TRB, et al (2021) Dcluster: Geospatial analytics with poi identification. Journal of Information and Data Management 12(2)

[5] Chen L, Thakuriah P, Ampountolas K (2021) Short-term prediction of demand for ride-hailing services: A deep learning approach. Journal of Big Data Analytics in Transportation 3:175–195

[6] Commission NTL (2023) Tlc trip record data. URL https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page

[7] Duckham M, Kulik L, Worboys M, et al (2008) Efficient generation of simple polygons for characterizing the shape of a set of points in the plane. Pattern recognition 41(10):3224–3236

[8] Edelsbrunner H, Kirkpatrick D, Seidel R (1983) On the shape of a set of points in the plane. IEEE Transactions on Information Theory 29(4):551–559. https://doi.org/10.1109/TIT.1983.1056714

[9] Ester M, Kriegel HP, Sander J, et al (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. In: kdd, pp 226–231

[10] Fanaee-T H (2013) Bike Sharing Dataset. UCI Machine Learning Repository, DOI: https://doi.org/10.24432/C5W894

[11] Graham RL (1972) An efficient algorithm for determining the convex hull of a finite planar set. Info Proc Lett 1:132–133

[12] Graser A (2019) MovingPandas: Efficient structures for movement data in Python. GI_Forum – Journal of Geographic Information Science 2019 7:54–68. https://doi.org/10.1553/giscience2019_01_s54

[13] Graser A (2023) movement-analysis-tools. URL https://github.com/anitagraser/movement-analysis-tools

[14] Haidri S, Haranwala YJ, Bogorny V, et al (2022) Ptrail—a python package for parallel trajectory data preprocessing. SoftwareX 19:101176

[15] Heo S, Lim CC, Bell ML (2020) Relationships between local green space and human mobility patterns during covid-19 for maryland and california, usa. Sustainability 12(22):9401

[16] Hu Y, Gao S, Janowicz K, et al (2015) Extracting and understanding urban areas of interest using geotagged photos. Computers, Environment and Urban Systems 54:240–254

[17] Jiang X, Bunke H (1998) Marked subgraph isomorphism of ordered graphs. In: Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR), Springer, pp 122–131

[18] Jiang X, Zheng C, Tian Y, et al (2015) Large-scale taxi o/d visual analytics for understanding metropolitan human movement patterns. Journal of Visualization 18:185–200

[19] Kashiyama T, Pang Y, Sekimoto Y, et al (2022) Pseudo-pflow: Development of nationwide synthetic open dataset for people movement based on limited travel survey and open statistical data. arXiv preprint arXiv:220500657

[20] Kovanen L, Karsai M, Kaski K, et al (2011) Temporal motifs in time-dependent networks. Journal of Statistical Mechanics: Theory and Experiment 2011(11):P11005

[21] Lee Y, Li Z, Li Y (2000) Taxonomy of space tessellation. ISPRS Journal of Photogrammetry and Remote Sensing 55(3):139–149. https://doi.org/https://doi.org/10.1016/S0924-2716(00)00015-0

[22] Lei D, Chen X, Cheng L, et al (2020) Inferring temporal motifs for travel pattern analysis using large scale smart card data. Transportation Research Part C: Emerging Technologies 120:102810

[23] Li M, Bao Z, Choudhury F, et al (2021) Aoi-shapes: An efficient footprint algorithm to support visualization of user-defined urban areas of interest. ACM Trans Interact Intell Syst 11(3–4). https://doi.org/10.1145/3431817, URL https://doi.org/10.1145/3431817

[24] Li R, He H, Wang R, et al (2021) Trajmesa: A distributed nosql-based trajectory data management system. IEEE Transactions on Knowledge and Data Engineering 35(1):1013–1027

[25] Luca M, Barlacchi G, Lepri B, et al (2021) A survey on deep learning for human mobility. ACM Computing Surveys (CSUR) 55(1):1–44

[26] Ma J, Li B, Mostafavi A (2022) Characterizing urban lifestyle signatures using motif properties in network of places. arXiv preprint arXiv:220401103

[27] Martin H, Hong Y, Wiedemann N, et al (2023) Trackintel: An open-source python library for human mobility analysis. Computers, Environment and Urban Systems 101:101938

[28] Montoliu R, Blom J, Gatica-Perez D (2013) Discovering places of interest in everyday life from smartphone data. Multimedia Tools And Applications 62(1):29. 179–207. https://doi.org/10.1007/s11042-011-0982-z

[29] Pappalardo L, Simini F, Rinzivillo S, et al (2015) Returners and explorers dichotomy in human mobility. Nature communications 6(1):1–8

[30] Pappalardo L, Simini F, Barlacchi G, et al (2021) scikit-mobility: a Python library for the analysis, generation and risk assessment of mobility data. arXiv:1907.07062

[31] Pérez-Torres R, Torres-Huitzil C, Galeana-Zapién H (2016) Full on-device stay points detection in smartphones for location-based mobile applications. Sensors 16(10):1693

[32] Psyllidis A, Gao S, Hu Y, et al (2022) Points of interest (poi): a commentary on the state of the art, challenges, and prospects for the future. Computational Urban Science 2(1):20

[33] Rizwan M, Wan W, Cervantes O, et al (2018) Using location-based social media data to observe check-in behavior and gender difference: Bringing weibo data into play. ISPRS International Journal of Geo-Information 7(5):196

[34] Santipantakis GM, Glenis A, Patroumpas K, et al (2020) Spartan: Semantic integration of big spatio-temporal data from streaming and archival sources. Future Generation Computer Systems 110:540–555

[35] Santos F, Almeida A, Martins C, et al (2019) Using poi functionality and accessibility levels for delivering personalized tourism recommendations. Computers, Environment and Urban Systems 77:101173

[36] Schneider CM, Belik V, Couronné T, et al (2013) Unravelling daily human mobility motifs. Journal of The Royal Society Interface 10(84):20130246

[37] Simini F, González MC, Maritan A, et al (2012) A universal model for mobility and migration patterns. Nature 484(7392):96–100

[38] Smolak K, Rohm W, Sila-Nowicka K (2022) Explaining human mobility predictions through a pattern matching algorithm. EPJ Data Science 11(1):45

[39] Su R, McBride EC, Goulias KG (2021) Unveiling daily activity pattern differences between telecommuters and commuters using human mobility motifs and sequence analysis. Transportation Research Part A: Policy and Practice 147:106–132

[40] Su R, Xiao J, McBride EC, et al (2021) Understanding senior's daily mobility patterns in california using human mobility motifs. Journal of transport geography 94:103117

[41] Sun X, Tan Y, Wu Q, et al (2019) Tm-miner: Tfs-based algorithm for mining temporal motifs in large temporal network. IEEE Access 7:49778–49789

[42] Xiong Q, Liu Y, Xie P, et al (2021) Revealing correlation patterns of individual location activity motifs between workdays and day-offs using massive mobile phone data. Computers, Environment and Urban Systems 89:101682

[43] Yabe T, Jones NK, Rao PSC, et al (2022) Mobile phone location data for disasters: A review from natural hazards and epidemics. Computers, Environment and Urban Systems 94:101777

[44] Yabe T, Tsubouchi K, Shimizu T, et al (2023) Metropolitan scale and longitudinal dataset of anonymized human mobility trajectories. arXiv preprint arXiv:230703401

[45] Yang X, Fang Z, Xu Y, et al (2021) Revealing temporal stay patterns in human mobility using large-scale mobile phone location data. Transactions in GIS 25(4):1927–1948

[46] Yang Y, Pentland A, Moro E (2023) Identifying latent activity behaviors and lifestyles using mobility data to describe urban dynamics. EPJ Data Science 12(1):15

[47] Yu J, Zhang Z, Sarwat M (2019) Spatial data management in Apache Spark: The GeoSpark perspective and beyond. Geoinformatica 23(1):37–78. https://doi.org/10.1007/s10707-018-0330-9

[48] Yu Q, Yuan J (2022) Transbigdata: A python package for transportation spatio-temporal big data processing, analysis and visualization. Journal of Open Source Software 7(71):4021

[49] Zaharia M, Chowdhury M, Das T, et al (2012) Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In: 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12). USENIX Association, San Jose, CA, pp 15–28

[50] Zheng VW, Zheng Y, Xie X, et al (2010) Collaborative location and activity recommendations with gps history data. In: Proceedings of the 19th international conference on World wide web, pp 1029–1038

[51] Zheng Y, Xie X, Ma WY, et al (2010) Geolife: A collaborative social networking service among user, location and trajectory. IEEE Data Eng Bull 33(2):32–39

[52] Zheng Y, Capra L, Wolfson O, et al (2014) Urban computing: Concepts, methodologies, and applications. ACM Transaction on Intelligent Systems and Technology